

시뮬레이터를 활용한 K-City Map 기반 자율주행 알고리즘 개발

프로젝트 지향 자율주행차 전문인력 양성과정

목차

1. K-CITY
2. GPS

1. K-CITY

• 자율주행 실험도시

- 경기도 화성에 위치
- 한국형 자율주행 실험 도시
- 고속도로, 도심, 교외, 주차시설 등의 도로 환경을 그대로 재현
- 각종 자율주행 자동차 실증 및 경진대회 개최지



K-CITY

- 자율주행 실험도시
 - 경기도 화성에 위치
 - 한국형 자율주행 실험 도시
 - 고속도로, 도심, 교외, 주차시설 등의 도로 환경을 그대로 재현
 - 각종 자율주행 자동차 실증 및 경진대회 개최지



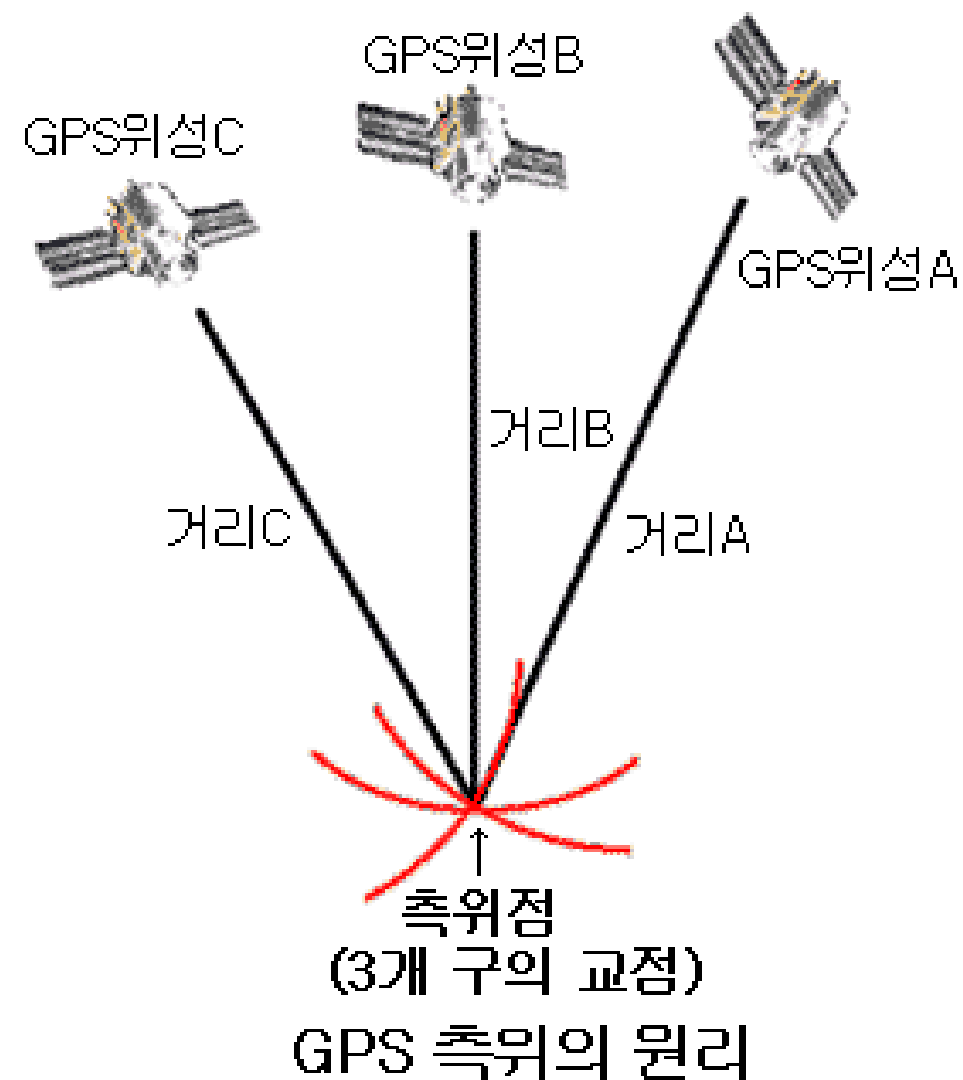
<시뮬레이터의 K-CITY 모습>

2. GPS

GPS

- **Global Positioning System**

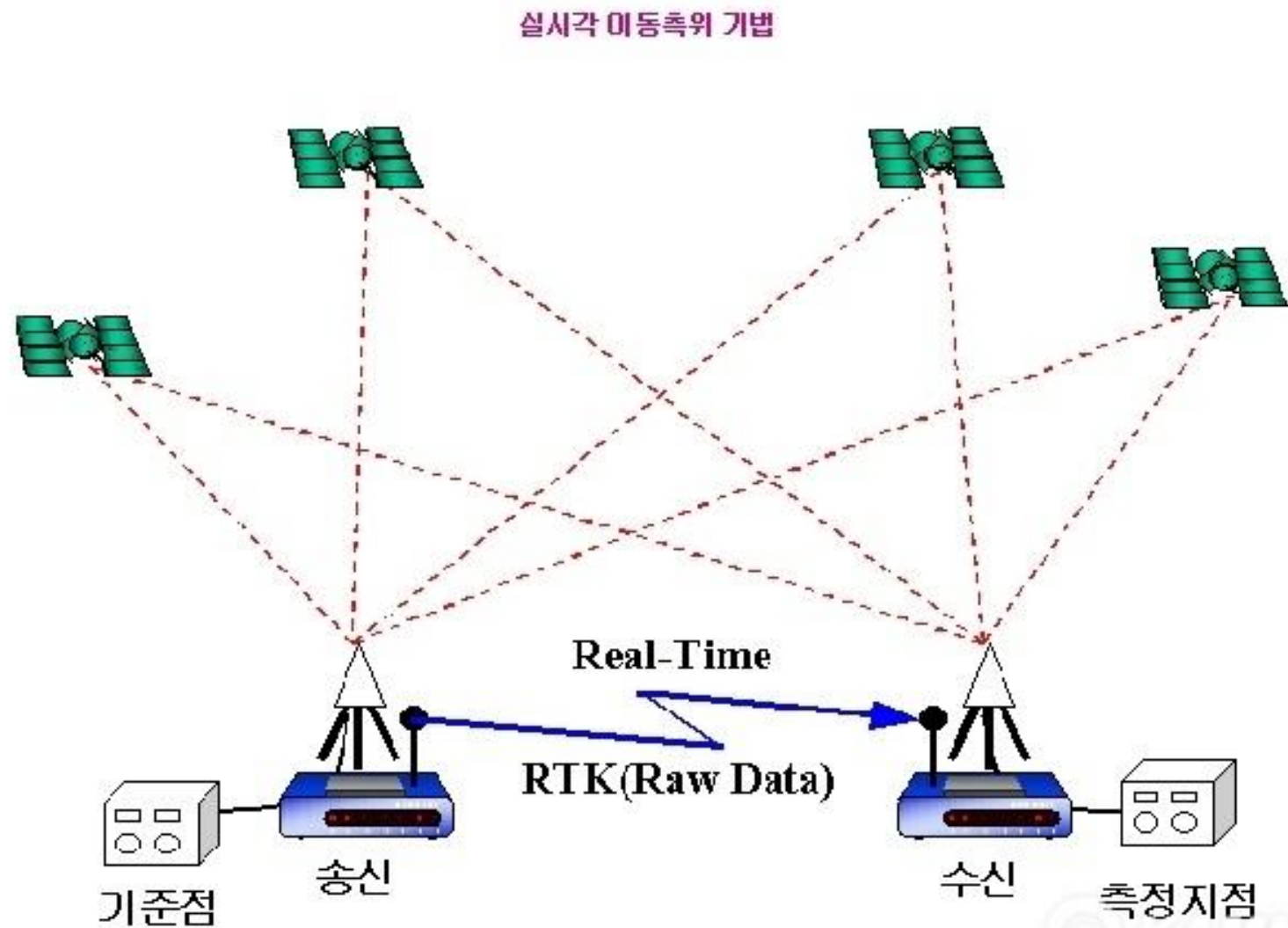
- 삼각측량의 원리를 이용함
- 위성은 정해진 궤도를 돌기 때문에, 각 위성간 위치와 각도는 이미 알고 있다.



GPS

- RTK

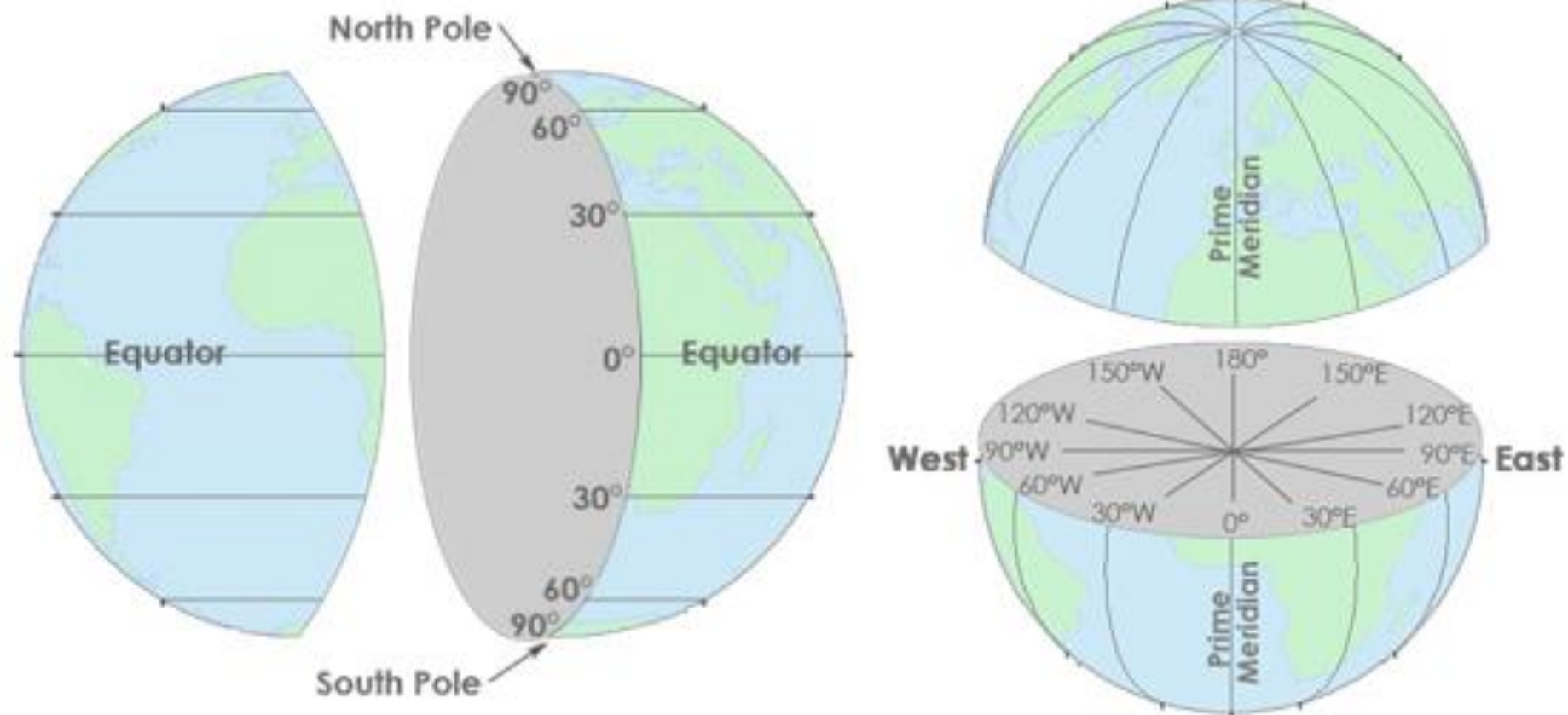
- 실시간 이동측위라는 개념으로 정밀한 위치정보를 가지고 있는 기준국의 반송파 위상에 대한 보정치를 이용
- 주로 LTE, DMB 신호를 이용함



GPS

- WGS84 좌표계

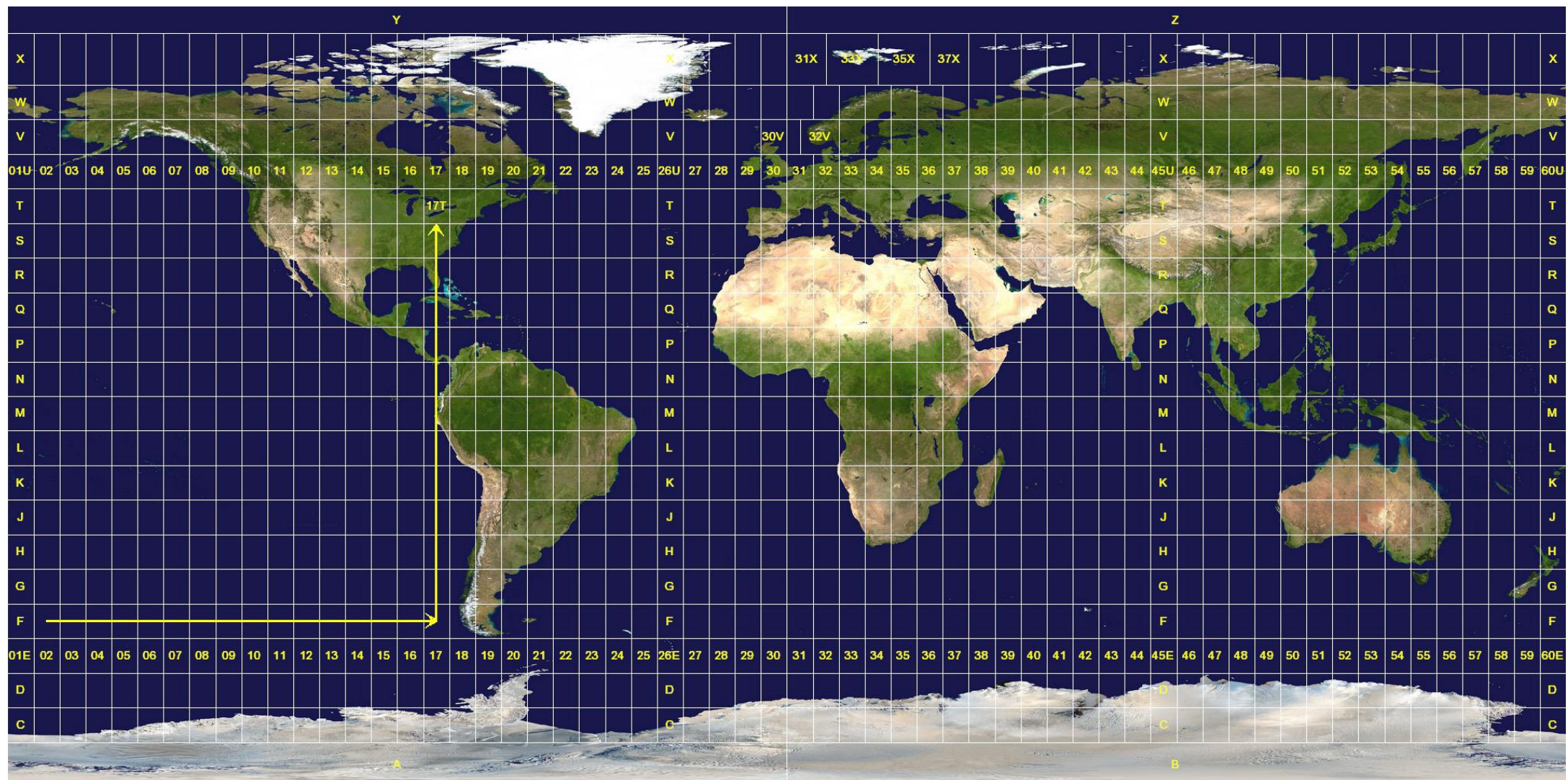
- 미국에서 군사용으로 GPS 시스템을 이용하면서 만든 타원체 모양의 좌표계
- 위도, 고도, 경도로 표현



GPS

• UTM 좌표

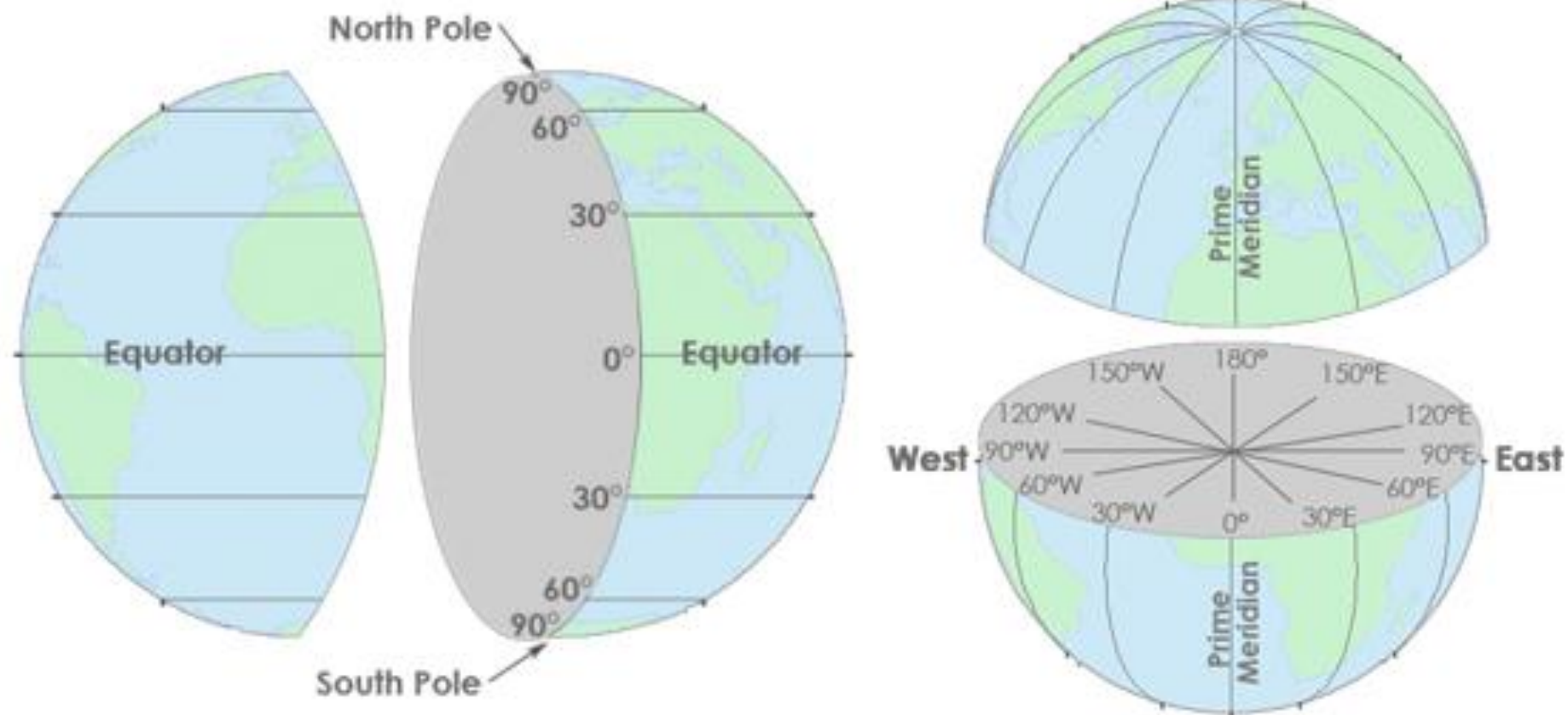
- WGS84 타원체는 3차원 좌표이므로 우리가 보는 2차원 지도상의 좌표와 다름
- 따라서 투영법을 이용해 곡면을 평면으로 바꾸어주는 과정이 필요
- UTM 좌표계는 평면 위의 임의의 점을 시발점으로 하여 그 점으로부터 XY 축으로 수직, 수평선을 긋고 각 축에 평행하게 격자망을 구성하여 사용
- 우리나라의 경우 52번째 Zone에 위치하며, 이 지역의 기준 원점인 경도 129도, 위도 0도를 UTM 좌표계의 원점으로 사용한다.



GPS

- WGS84 좌표계

- 미국에서 군사용으로 GPS 시스템을 이용하면서 만든 타원체 모양의 좌표계
- 위도, 고도, 경도로 표현



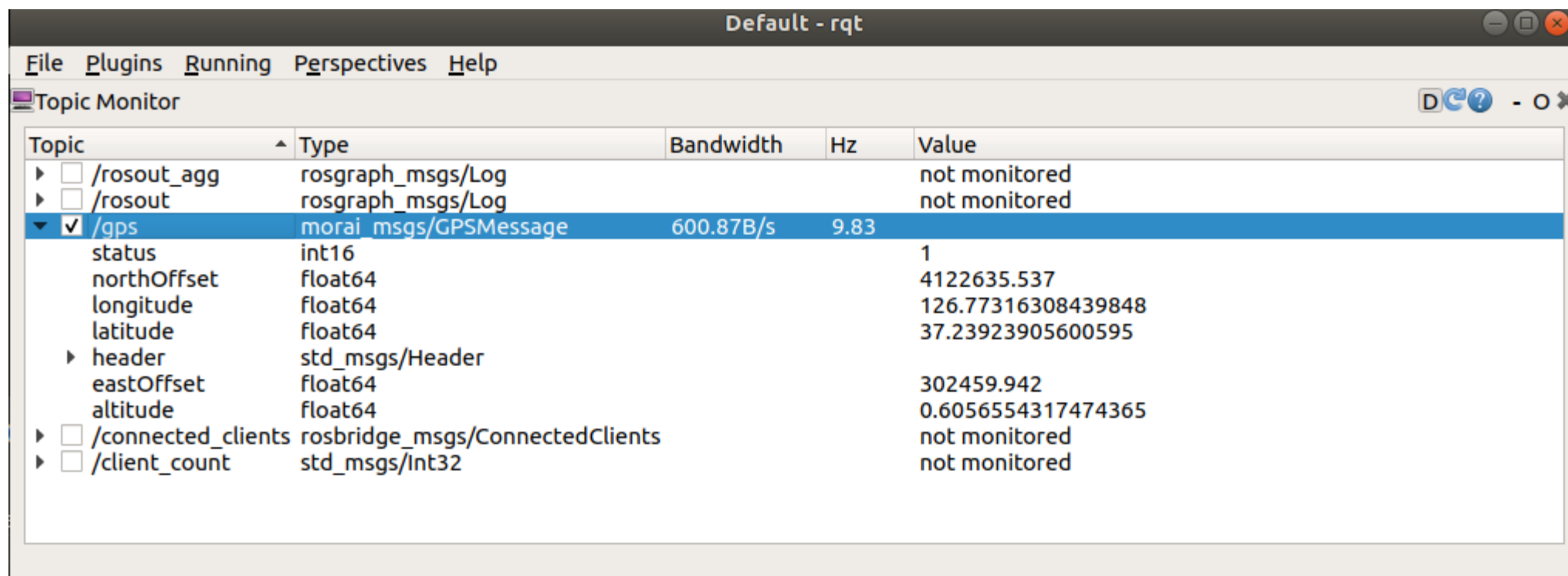
GPS

- 실습
 - Wecar에 GPS 센서를 부착
 - 위도, 경도, 고도 값을 알 수 있다.
 - rqt topic 모니터를 이용해 /gps 토픽 확인.



GPS

- 실습
 - Wecar에 GPS 센서를 부착
 - 위도, 경도, 고도 값을 알 수 있다.
 - rqt topic 모니터를 이용해 /gps 토픽 확인.



The screenshot shows the 'Default - rqt' window with the 'Topic Monitor' tab selected. The table below represents the data shown in the monitor.

Topic	Type	Bandwidth	Hz	Value
<input type="checkbox"/> /rosout_agg	rosgraph_msgs/Log			not monitored
<input type="checkbox"/> /rosout	rosgraph_msgs/Log			not monitored
<input checked="" type="checkbox"/> /gps	morai_msgs/GPSMessage	600.87B/s	9.83	
status	int16			1
northOffset	float64			4122635.537
longitude	float64			126.77316308439848
latitude	float64			37.23923905600595
▶ header	std_msgs/Header			
eastOffset	float64			302459.942
altitude	float64			0.6056554317474365
<input type="checkbox"/> /connected_clients	rosbridge_msgs/ConnectedClients			not monitored
<input type="checkbox"/> /client_count	std_msgs/Int32			not monitored

GPS

- 실습(방법1)
 - ros 패키지 이용
 - 이미 만들어져 있는 오픈소스 사용

gps_common

kinetic melodic **noetic** Show EOL distros: ☐

[Documentation Status](#)

[gps_umd](#): [gps_common](#) | [gpsd_client](#)

Package Summary

✓ Released ✓ Continuous Integration ✓ Documented

GPS messages and common routines for use in GPS drivers

- Maintainer status: maintained
- Maintainer: Timo Roehling <timo.roehling AT fkie.fraunhofer DOT de>, P. J. Reed <preed AT swri DOT org>
- Author:
- License: BSD
- Source: git https://github.com/swri-robotics/gps_umd.git (branch: master)

차례

1. Experimental Package
2. Messages
3. Nodes
 1. utm_odometry_node
 1. Subscribed Topics
 2. Published Topics
 3. Parameters
4. Tutorial

Package Links

[Code API](#)
[Msg API](#)
[FAQ](#)
[Changelog](#)
[Change List](#)
[Reviews](#)

[Dependencies](#) (9)
[Used by](#) (2)
[Jenkins jobs](#) (10)

2. Messages

gps_common defines two common messages for GPS drivers to output: [gps_common/GPSFix](#) and [gps_common/GPSStatus](#).

In most cases, these messages should be published simultaneously, with identical timestamps.

3. Nodes

3.1 utm_odometry_node

utm_odometry_node converts latitude-longitude readings into UTM odometry

3.1.1 Subscribed Topics

fix ([sensor_msgs/NavSatFix](#))
GPS measurement and status

3.1.2 Published Topics

odom ([nav_msgs/Odometry](#))
UTM-encoded position

3.1.3 Parameters

~rot_covariance (double, default: 99999)

Variance (in meters) to specify for rotational measurements

~frame_id (string, default: Copy frame_id from fix message)

Frame to specify in header of outgoing Odometry message

~child_frame_id (string)

Child frame to specify in header of outgoing Odometry message

4. Tutorial

See [gpsd_client](#) for an example of a sender node that uses this package's messages.

1. Experimental Package

This package is a space to stage messages and common GPS-processing routines that are undergoing a standardization process. Its contents will probably be moved into [ros-pkg](#) once they've matured.

GPS

- 실습(방법2)
 - 직접 변환하는 코드를 작성

```
1  #!/usr/bin/env python
2  import rospy
3  import numpy as np
4  import tf
5
6  from sensor_msgs.msg import NavSatFix
7  from std_msgs.msg import Int16
8  from nav_msgs.msg import Odometry
9  from morai_msgs.msg import GPSTime
10 from tf.transformations import quaternion_from_euler
11 from geometry_msgs.msg import PoseStamped, Quaternion, TwistStamped
12 from math import atan2, pow, sqrt, pi
13
14 def proj_coef_0(e):
15     c0_transverse_mercator = np.array([
16         [-175 / 16384.0, 0.0, -5 / 2560.0, 0.0, -3 / 64.0, 0.0, -1 / 4.0, 0.0, 1.0],
17         [-105 / 40960.0, 0.0, -45 / 1024.0, 0.0, -3 / 32.0, 0.0, -3 / 8.0, 0.0, 0.0],
18         [ 525 / 16384.0, 0.0, 45 / 1024.0, 0.0, 15 / 256.0, 0.0, 0.0, 0.0, 0.0],
19         [-175 / 12288.0, 0.0, -35 / 3072.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
20         [ 315 / 131072.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
21     ])
22
23     c_out = np.zeros(5)
24
25     for i in range(0,5):
26         c_out[i] = np.poly1d(c0_transverse_mercator[i,:])(e)
27
28     return c_out
29
30
31 def proj_coef_2(e):
32     c0_meridian_arc = np.array([
33         [-175 / 16384.0, 0.0, -5 / 256.0, 0.0, -3 / 64.0, 0.0, -1 / 4.0, 0.0, 1.0 ],
34         [-901 / 184320.0, 0.0, -9 / 1024.0, 0.0, -1 / 96.0, 0.0, 1 / 8.0, 0.0, 0.0 ],
35         [-311 / 737280.0, 0.0, 17 / 5120.0, 0.0, 13 / 768.0, 0.0, 0.0, 0.0, 0.0 ],
36         [ 899 / 430080.0, 0.0, 61 / 15360.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ],
37         [ 49561 / 41287680.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ]
38     ])
39
40     c_out = np.zeros(5)
```


GPS

- 실습(방법2)
 - 직접 변환하는 코드를 작성

```
40     c_out = np.zeros(5)
41
42     for i in range(0,5):
43         c_out[i] = np.poly1d(c0_meridian_arc[i,:])(e)
44
45     return c_out
46
47 class LocationSensor:
48     def __init__(self, zone=52):
49         self.gps_sub = rospy.Subscriber("/gps", GPSTMessage, self.navsat_callback)
50         self.odom_pub = rospy.Publisher("/morai_odom", Odometry, queue_size=1)
51
52         self.zone = zone
53         self.vel=0
54         self.vehicle_p = PoseStamped()
55         self.prev_x=0
56         self.prev_y=0
57         self.prev_heading=0
58         # rad to deg
59         self.D0 = 180 / np.pi
60
61         # WGS84
62         self.A1 = 6378137.0
63         self.F1 = 298.257223563
64
65         # Scale Factor
66         self.K0 = 0.9996
67
68         # False East & North
69         self.X0 = 500000
70         if (self.zone > 0):
71             self.Y0 = 0.0
72         else:
73             self.Y0 = 1e7
74
75         # UTM origin latitude & longitude
76         self.P0 = 0 / self.D0
77         self.L0 = (6 * abs(self.zone) - 183) / self.D0
78
79         # ellipsoid eccentricity
80         self.B1 = self.A1 * (1 - 1 / self.F1)
```

GPS

- 실습(방법2)
 - 직접 변환하는 코드를 작성

```
80 self.B1 = self.A1 * (1 - 1 / self.F1)
81 self.E1 = np.sqrt((self.A1**2 - self.B1**2) / (self.A1**2))
82 self.N = self.K0 * self.A1
83
84 # mercator transverse proj params
85 self.C = np.zeros(5)
86 self.C = proj_coef_0(self.E1)
87
88 self.YS = self.Y0 - self.N * (
89     self.C[0] * self.P0
90     + self.C[1] * np.sin(2 * self.P0)
91     + self.C[2] * np.sin(4 * self.P0)
92     + self.C[3] * np.sin(6 * self.P0)
93     + self.C[4] * np.sin(8 * self.P0))
94
95 self.C2 = proj_coef_2(self.E1)
96
97 self.rate = rospy.Rate(30)
98
99 self.x, self.y, self.heading, self.velocity, self.gps_status = None, None, None, None, None
100
101 self.x_old, self.y_old = 0, 0
102
103
104 def convertLL2UTM(self, lat, lon):
105
106     p1 = lat / self.D0 # Phi = Latitude(rad)
107     l1 = lon / self.D0 # Lambda = Longitude(rad)
108
109     es = self.E1 * np.sin(p1)
110     L = np.log( np.tan(np.pi/4.0 + p1/2.0) *
111                np.power( ((1 - es) / (1 + es)), (self.E1 / 2)))
112
113     z = np.complex(
114         np.arctan(np.sinh(L) / np.cos(l1 - self.L0)),
115         np.log(np.tan(np.pi / 4.0 + np.arcsin(np.sin(l1 - self.L0) / np.cosh(L)) / 2.0))
116     )
117
118     Z = self.N * self.C2[0] * z \
119         + self.N * (self.C2[1] * np.sin(2.0 * z)
120         + self.C2[2] * np.sin(4.0 * z)
```

GPS

- 실습(방법2)
 - 직접 변환하는 코드를 작성

```
120         + self.C2[2] * np.sin(4.0 * z)
121         + self.C2[3] * np.sin(6.0 * z)
122         + self.C2[4] * np.sin(8.0 * z))
123
124     east = Z.imag + self.X0
125     north = Z.real + self.YS
126
127     return east, north
128
129     def navsat_callback(self, gps_msg):
130
131         lat = gps_msg.latitude
132         lon = gps_msg.longitude
133
134         e_o = gps_msg.eastOffset
135         n_o = gps_msg.northOffset
136
137         e_global, n_global = self.convertLL2UTM(lat, lon)
138
139         x,y = e_global - e_o, n_global - n_o
140
141         odom_msg=Odometry()
142         odom_msg.child_frame_id='base_link'
143         odom_msg.header.frame_id='map'
144         odom_msg.header.stamp=rospy.Time.now()
145         odom_msg.pose.pose.position.x=x
146         odom_msg.pose.pose.position.y=y
147         odom_msg.pose.pose.position.z=0
148         odom_msg.pose.pose.orientation.x=0
149         odom_msg.pose.pose.orientation.y=0
150         odom_msg.pose.pose.orientation.z=0
151         odom_msg.pose.pose.orientation.w=1
152         self.odom_pub.publish(odom_msg)
153
```

GPS

- 실습(방법2)
 - 직접 변환하는 코드를 작성

```
120         + self.C2[2] * np.sin(4.0 * z)
121         + self.C2[3] * np.sin(6.0 * z)
122         + self.C2[4] * np.sin(8.0 * z))
123
124     east = Z.imag + self.X0
125     north = Z.real + self.YS
126
127     return east, north
128
129     def navsat_callback(self, gps_msg):
130
131         lat = gps_msg.latitude
132         lon = gps_msg.longitude
133
134         e_o = gps_msg.eastOffset
135         n_o = gps_msg.northOffset
136
137         e_global, n_global = self.convertLL2UTM(lat, lon)
138
139         x,y = e_global - e_o, n_global - n_o
140
141         odom_msg=Odometry()
142         odom_msg.child_frame_id='base_link'
143         odom_msg.header.frame_id='map'
144         odom_msg.header.stamp=rospy.Time.now()
145         odom_msg.pose.pose.position.x=x
146         odom_msg.pose.pose.position.y=y
147         odom_msg.pose.pose.position.z=0
148         odom_msg.pose.pose.orientation.x=0
149         odom_msg.pose.pose.orientation.y=0
150         odom_msg.pose.pose.orientation.z=0
151         odom_msg.pose.pose.orientation.w=1
152         self.odom_pub.publish(odom_msg)
153
154
155
156     if __name__ == '__main__':
157
158         rospy.init_node('gps_parser', anonymous=True)
159
160         loc_sensor = LocationSensor()
161         rospy.spin()
```

GPS

- 변환 검증 방법
 - 가지고 있는 정밀지도 데이터와 비교
 - LL2UTM 변환 사이트에 값을 넣어서 확인
<https://www.latlong.net/lat-long-utm.html>

Convert Lat Long to UTM

This is an effective and fast online *Lat Long to UTM converter*. It can be used to make the stated conversions at any time and any place. Type the latitude and longitude values to convert from lat long coordinate system into **UTM** (Universal Transverse Mercator) coordinate system.

Type the latitude and longitude values to convert into **UTM**
(Universal Transverse Mercator) coordinate system.

Latitude	Longitude	Convert
0.000000	0.000000	
UTM Easting	UTM Northing	UTM Zone
UTM Easting	UTM Northing	UTM Zone

END